zqatnqld1

April 17, 2024

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
df1 = {
'Name':['George','Andrea','micheal','maggie','Ravi','Xien','Jalpa',np.nan],
'State':
 ↪['Arizona','Georgiaprint(df1)print(df1)','Newyork','Indiana','Florida','California',np.
 ↪nan,np.nan],
'Gender':["M","F","M","F","M","M",np.nan,np.nan],
'Score':[63,48,56,75,np.nan,77,np.nan,np.nan]
}
```

[4]:
```
df1 = pd.DataFrame(df1,columns=['Name','State','Gender','Score'])
```

[5]:
```
print(df1)
```

```
      Name                         State Gender  Score
0   George                       Arizona      M   63.0
1   Andrea  Georgiaprint(df1)print(df1)      F   48.0
2  micheal                       Newyork      M   56.0
3   maggie                       Indiana      F   75.0
4     Ravi                       Florida      M    NaN
5     Xien                    California      M   77.0
6    Jalpa                           NaN    NaN    NaN
```

[6]:
```
df1.shape
```

[6]:
```
(7, 4)
```

[7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7 entries, 0 to 6
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    7 non-null      object
 1   State   6 non-null      object
```

```
 2   Gender  6 non-null      object
 3   Score   5 non-null      float64
dtypes: float64(1), object(3)
memory usage: 280.0+ bytes
```

[8]:
```
df1.describe()
```

[8]:
```
           Score
count   5.000000
mean   63.800000
std    12.357184
min    48.000000
25%    56.000000
50%    63.000000
75%    75.000000
max    77.000000
```

[9]:
```
df1.isnull()
```

[9]:
```
    Name  State  Gender  Score
0  False  False   False  False
1  False  False   False  False
2  False  False   False  False
3  False  False   False  False
4  False  False   False   True
5  False  False   False  False
6  False   True    True   True
```

[10]:
```
df1.isnull().sum()
```

[10]:
```
Name      0
State     1
Gender    1
Score     2
dtype: int64
```

[11]:
```
df1 = df1.dropna(subset=["Name"])
```

[12]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7 entries, 0 to 6
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    7 non-null      object
 1   State   6 non-null      object
 2   Gender  6 non-null      object
```

```
 3   Score   5 non-null      float64
dtypes: float64(1), object(3)
memory usage: 280.0+ bytes
```

[13]:
```
df1.head(2)
```

[13]:
```
      Name                         State Gender  Score
0   George                       Arizona      M   63.0
1   Andrea  Georgiaprint(df1)print(df1)      F   48.0
```

[14]:
```
df1.select_dtypes(exclude='object').dtypes
```

[14]:
```
Score    float64
dtype: object
```

[15]:
```
df1.select_dtypes(include='object').dtypes
```

[15]:
```
Name      object
State     object
Gender    object
dtype: object
```

[16]:
```
df1.tail()
```

[16]:
```
       Name       State Gender  Score
2  micheal     Newyork      M   56.0
3   maggie     Indiana      F   75.0
4     Ravi     Florida      M    NaN
5     Xien  California      M   77.0
6    Jalpa         NaN    NaN    NaN
```

is2j8ivg5

April 17, 2024

[1]:
```
import numpy as np
import pandas as pd
```

[14]:
```
df = pd.read_csv("xAPI-Edu-Data.csv")
```

[15]:
```
df
```

[15]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion	ParentAnsweringSurvey
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	2	20	Yes
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	3	25	Yes
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	0	30	No
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	5	35	No
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	12	50	No
..	…	…	…	…	…	…	…	…	…	…	…	…	…	…
475	F	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	S	Father	5	4	1	10	No
476	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	F	Father	50	77	2	77	Yes
477	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	S	Father	55	74	1	70	Yes
478	F	Jordan	Jordan	MiddleSchool	G-08	A	History	F	Father	30	17	0	20	No
479	F	Jordan	Jordan	MiddleSchool	G-08	A	History	S	Father	35	14	0	15	No

```
3                    5          35                    No
4                   12          50                    No
..                  …           …                     …
475                  5           8                    No
476                 14          28                    No
477                 25          29                    No
478                 14          57                    No
479                 23          62                    No

    ParentschoolSatisfaction StudentAbsenceDays Class
0                       Good            Under-7     M
1                       Good            Under-7     M
2                        Bad            Above-7     L
3                        Bad            Above-7     L
4                        Bad            Above-7     M
..                       …                 …   …
475                      Bad            Above-7     L
476                      Bad            Under-7     M
477                      Bad            Under-7     M
478                      Bad            Above-7     L
479                      Bad            Above-7     L

[480 rows x 17 columns]
```

[16]:
```
df.head()
```

[16]:
```
  gender NationalITy PlaceofBirth     StageID GradeID SectionID Topic  \
0      M          KW       KuwaIT  lowerlevel    G-04         A    IT
1      M          KW       KuwaIT  lowerlevel    G-04         A    IT
2      M          KW       KuwaIT  lowerlevel    G-04         A    IT
3      M          KW       KuwaIT  lowerlevel    G-04         A    IT
4      M          KW       KuwaIT  lowerlevel    G-04         A    IT

  Semester Relation  raisedhands  VisITedResources  AnnouncementsView  \
0        F   Father           15                16                  2
1        F   Father           20                20                  3
2        F   Father           10                 7                  0
3        F   Father           30                25                  5
4        F   Father           40                50                 12

   Discussion ParentAnsweringSurvey ParentschoolSatisfaction  \
0          20                   Yes                     Good
1          25                   Yes                     Good
2          30                    No                      Bad
3          35                    No                      Bad
4          50                    No                      Bad
```

```
  StudentAbsenceDays Class
0            Under-7     M
1            Under-7     M
2            Above-7     L
3            Above-7     L
4            Above-7     M
```

[17]:
```
df.tail()
```

[17]:
```
    gender NationalITy PlaceofBirth       StageID GradeID SectionID  \
475      F      Jordan       Jordan  MiddleSchool    G-08         A
476      F      Jordan       Jordan  MiddleSchool    G-08         A
477      F      Jordan       Jordan  MiddleSchool    G-08         A
478      F      Jordan       Jordan  MiddleSchool    G-08         A
479      F      Jordan       Jordan  MiddleSchool    G-08         A

         Topic Semester Relation  raisedhands  VisITedResources  \
475  Chemistry        S   Father            5                 4
476    Geology        F   Father           50                77
477    Geology        S   Father           55                74
478    History        F   Father           30                17
479    History        S   Father           35                14

     AnnouncementsView  Discussion ParentAnsweringSurvey  \
475                  5           8                    No
476                 14          28                    No
477                 25          29                    No
478                 14          57                    No
479                 23          62                    No

    ParentschoolSatisfaction StudentAbsenceDays Class
475                      Bad            Above-7     L
476                      Bad            Under-7     M
477                      Bad            Under-7     M
478                      Bad            Above-7     L
479                      Bad            Above-7     L
```

[18]:
```
df.describe()
```

[18]:
```
       raisedhands  VisITedResources  AnnouncementsView  Discussion
count   480.000000        480.000000         480.000000  480.000000
mean     46.775000         54.797917          37.918750   43.283333
std      30.779223         33.080007          26.611244   27.637735
min       0.000000          0.000000           0.000000    1.000000
25%      15.750000         20.000000          14.000000   20.000000
50%      50.000000         65.000000          33.000000   39.000000
75%      75.000000         84.000000          58.000000   70.000000
```

```
max     100.000000          99.000000           98.000000    99.000000
```

[19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   gender                    480 non-null    object
 1   NationalITy               480 non-null    object
 2   PlaceofBirth              480 non-null    object
 3   StageID                   480 non-null    object
 4   GradeID                   480 non-null    object
 5   SectionID                 480 non-null    object
 6   Topic                     480 non-null    object
 7   Semester                  480 non-null    object
 8   Relation                  480 non-null    object
 9   raisedhands               480 non-null    int64
 10  VisITedResources          480 non-null    int64
 11  AnnouncementsView         480 non-null    int64
 12  Discussion                480 non-null    int64
 13  ParentAnsweringSurvey     480 non-null    object
 14  ParentschoolSatisfaction  480 non-null    object
 15  StudentAbsenceDays        480 non-null    object
 16  Class                     480 non-null    object
dtypes: int64(4), object(13)
memory usage: 63.9+ KB
```

[20]:
```
df.shape
```

[20]:
```
(480, 17)
```

[21]:
```
df.isnull().any().any()
```

[21]:
```
False
```

[22]:
```
df.isnull().sum()
```

[22]:
```
gender                    0
NationalITy               0
PlaceofBirth              0
StageID                   0
GradeID                   0
SectionID                 0
Topic                     0
Semester                  0
```

```
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64
```

[23]:
```
avg_val = df["Discussion"].astype("float").mean()
avg_val
```

[23]:
```
43.28333333333333
```

[24]:
```
df["Discussion"].replace(np.NaN, avg_val, inplace=False)
```

[24]:
```
0      20
1      25
2      30
3      35
4      50
       ..
475     8
476    28
477    29
478    57
479    62
Name: Discussion, Length: 480, dtype: int64
```

[25]:
```
df.isnull().sum()
```

[25]:
```
gender                      0
NationalITy                 0
PlaceofBirth                0
StageID                     0
GradeID                     0
SectionID                   0
Topic                       0
Semester                    0
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
```

```
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64
```

1fqrlpvmk

April 17, 2024

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

[1]:
```
import numpy as np
import pandas as pd
import statistics as st
```

[2]:
```
df = pd.read_csv("Mall_Customers.csv")
```

[3]:
```
df
```

[3]:
```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0             1    Male   19                  15                      39
1             2    Male   21                  15                      81
2             3  Female   20                  16                       6
3             4  Female   23                  16                      77
4             5  Female   31                  17                      40
..          …     …  …                   …                       …
195         196  Female   35                 120                      79
196         197  Female   45                 126                      28
197         198    Male   32                 126                      74
198         199    Male   32                 137                      18
199         200    Male   30                 137                      83

[200 rows x 5 columns]
```

[37]:
```
df.mean
```

[37]:
```
<bound method DataFrame.mean of      CustomerID  Gender  Age  Annual Income (k$)
Spending Score (1-100)
0             1    Male   19                  15                      39
1             2    Male   21                  15                      81
2             3  Female   20                  16                       6
3             4  Female   23                  16                      77
4             5  Female   31                  17                      40
```

```
..          …     …  …                 …                      …
195         196  Female   35                 120                    79
196         197  Female   45                 126                    28
197         198    Male   32                 126                    74
198         199    Male   32                 137                    18
199         200    Male   30                 137                    83

[200 rows x 5 columns]>
```

[38]:
```
df.loc[:,'Age'].mean
```

[38]:
```
<bound method Series.mean of 0      19
1      21
2      20
3      23
4      31
       ..
195    35
196    45
197    32
198    32
199    30
Name: Age, Length: 200, dtype: int64>
```

[40]:
```
df.median
```

[40]:
```
<bound method DataFrame.median of      CustomerID  Gender  Age  Annual Income
(k$)  Spending Score (1-100)
0             1    Male   19                 15                    39
1             2    Male   21                 15                    81
2             3  Female   20                 16                     6
3             4  Female   23                 16                    77
4             5  Female   31                 17                    40
..          …     …  …                 …                      …
195         196  Female   35                 120                    79
196         197  Female   45                 126                    28
197         198    Male   32                 126                    74
198         199    Male   32                 137                    18
199         200    Male   30                 137                    83

[200 rows x 5 columns]>
```

[41]:
```
df.loc[:,'Age'].median()
```

[41]:
```
36.0
```

[43]:
```
df.mode()
```

[43]:
```
     CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-100)
0             1  Female  32.0                54.0                    42.0
1             2     NaN   NaN                78.0                     NaN
2             3     NaN   NaN                 NaN                     NaN
3             4     NaN   NaN                 NaN                     NaN
4             5     NaN   NaN                 NaN                     NaN
..          ...     ...   ...                 ...                     ...
195         196     NaN   NaN                 NaN                     NaN
196         197     NaN   NaN                 NaN                     NaN
197         198     NaN   NaN                 NaN                     NaN
198         199     NaN   NaN                 NaN                     NaN
199         200     NaN   NaN                 NaN                     NaN

[200 rows x 5 columns]
```

[44]:
```
df.loc[:,'Age'].mode()
```

[44]:
```
0    32
Name: Age, dtype: int64
```

[45]:
```
df.min()
```

[45]:
```
CustomerID                    1
Gender                   Female
Age                          18
Annual Income (k$)           15
Spending Score (1-100)        1
dtype: object
```

[46]:
```
df.loc[:,'Age'].min(skipna = False)
```

[46]:
```
18
```

[47]:
```
df.max()
```

[47]:
```
CustomerID                  200
Gender                     Male
Age                          70
Annual Income (k$)          137
Spending Score (1-100)       99
dtype: object
```

[48]:
```
df.loc[:,'Age'].max(skipna = False)
```

[48]:
```
70
```

[49]:
```
df.std
```

[49]:
```
<bound method DataFrame.std of      CustomerID  Gender  Age  Annual Income (k$)
Spending Score (1-100)
0              1    Male   19                  15                      39
1              2    Male   21                  15                      81
2              3  Female   20                  16                       6
3              4  Female   23                  16                      77
4              5  Female   31                  17                      40
..           …     …  …                   …                       …
195          196  Female   35                 120                      79
196          197  Female   45                 126                      28
197          198    Male   32                 126                      74
198          199    Male   32                 137                      18
199          200    Male   30                 137                      83

[200 rows x 5 columns]>
```

[50]:
```
df.loc[:,'Age'].std()
```

[50]:
```
13.96900733155888
```

[54]:
```
df.groupby(['Gender'])['Age'].mean()
```

[54]:
```
Gender
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

[55]:
```
df_u=df.rename(columns= {'Annual Income (k$)':'Income'}, inplace= False)
```

[56]:
```
df_u
```

[56]:
```
     CustomerID  Gender  Age  Income  Spending Score (1-100)
0             1    Male   19      15                      39
1             2    Male   21      15                      81
2             3  Female   20      16                       6
3             4  Female   23      16                      77
4             5  Female   31      17                      40
..          …     …  …     …                       …
195         196  Female   35     120                      79
196         197  Female   45     126                      28
197         198    Male   32     126                      74
198         199    Male   32     137                      18
199         200    Male   30     137                      83

[200 rows x 5 columns]
```

[58]:
```
df_u.groupby(['Gender']).Income.mean()
```

[58]:
```
Gender
Female    59.250000
Male      62.227273
Name: Income, dtype: float64
```

[61]:
```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['Gender']]).toarray())
enc_df

df_encode =df_u.join(enc_df)
df_encode
```

[61]:

	CustomerID	Gender	Age	Income	Spending Score (1-100)	0	1
0	1	Male	19	15	39	0.0	1.0
1	2	Male	21	15	81	0.0	1.0
2	3	Female	20	16	6	1.0	0.0
3	4	Female	23	16	77	1.0	0.0
4	5	Female	31	17	40	1.0	0.0
..	…	…	…	…	…	…	…
195	196	Female	35	120	79	1.0	0.0
196	197	Female	45	126	28	1.0	0.0
197	198	Male	32	126	74	0.0	1.0
198	199	Male	32	137	18	0.0	1.0
199	200	Male	30	137	83	0.0	1.0

[200 rows x 7 columns]

2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset. Provide the codes with outputs and explain everything that you do in this step.

[62]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[63]:
```
df_iris = pd.read_csv("Iris.csv")
df_iris.head()
```

[63]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

[66]:
```
print('Iris-setosa')
setosa = df_iris['variety'] == 'Iris-setosa'
print(df_iris[setosa].describe())
print('\nIris-versicolor')
versicolor = df_iris['variety'] == 'Iris-versicolor'
print(df_iris[versicolor].describe())
print('\nIris-virginica')
virginica = df_iris['variety'] == 'Iris-virginica'
print(df_iris[virginica].describe())
```

```
Iris-setosa
       sepal.length  sepal.width  petal.length  petal.width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN

Iris-versicolor
       sepal.length  sepal.width  petal.length  petal.width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN

Iris-virginica
       sepal.length  sepal.width  petal.length  petal.width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN
```

[67]:
```
df_iris.dtypes.value_counts()
```

[67]:
```
float64    4
object     1
```

```
Name: count, dtype: int64
```

practical4

April 19, 2024

[6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[18]:
```
data = pd.read_csv("boston.csv")
```

[19]:
```
data.head()
```

[19]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

[33]:
```
X=data['dis'].values
Y=data['medv'].values
```

[34]:
```
Y
```

[34]:
```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
```

```
21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
 7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
 8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
 9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9])
```

[35]:
```
mean_x=np.mean(X)
mean_y=np.mean(Y)
n=len(X)
n
```

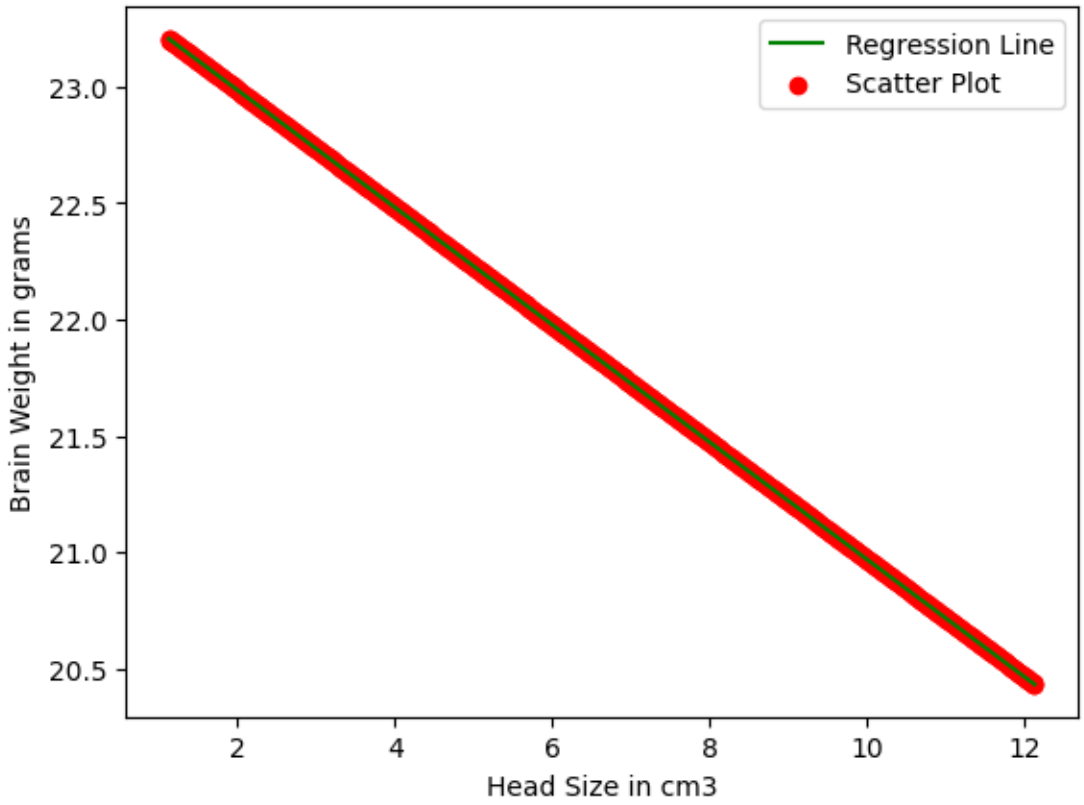
[35]: 506

[36]:
```
numer=0
denom=0
for i in range(n):
```

```
    numer+=(X[i]-mean_x)*(y[i]-mean_y)
    denom+=(X[i]-mean_x)**2
b1=numer/denom
b0=mean_y-(b1*mean_x)
print("Coeeficients")
print("m=",b1)
print("c=",b0)
```

```
Coeeficients
m= -0.2521585687373985
c= 23.4897588565503
```

[40]:
```
max_x=np.max(X)
min_x=np.min(X)
x=np.linspace(min_x,max_x,1000)
y=b0+b1*x
plt.plot(x,y,color='green',label='Regression Line')
plt.scatter(X,Y,c='red',label='Scatter Plot')
plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')
plt.legend()
plt.show()
```



al0cifsb8

April 17, 2024

[12]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[13]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

[14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
 ↪random_state = 0)
```

[15]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[16]:
```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state = 0)
log_reg.fit(X_train, y_train)
```

[16]:
```
LogisticRegression(random_state=0)
```

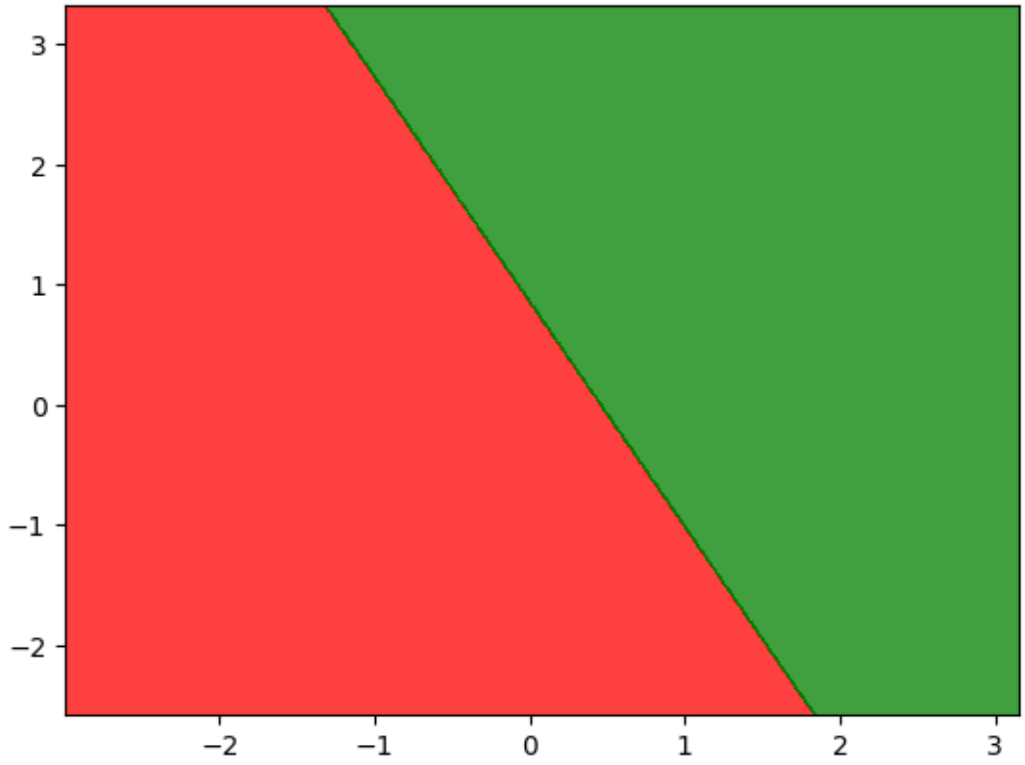
[17]:
```
y_pred = log_reg.predict(X_test)
```

[18]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

[19]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
 ↪0].max() + 1, step = 0.01),                      np.arange(start = X_set[:,␣
 ↪1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
```

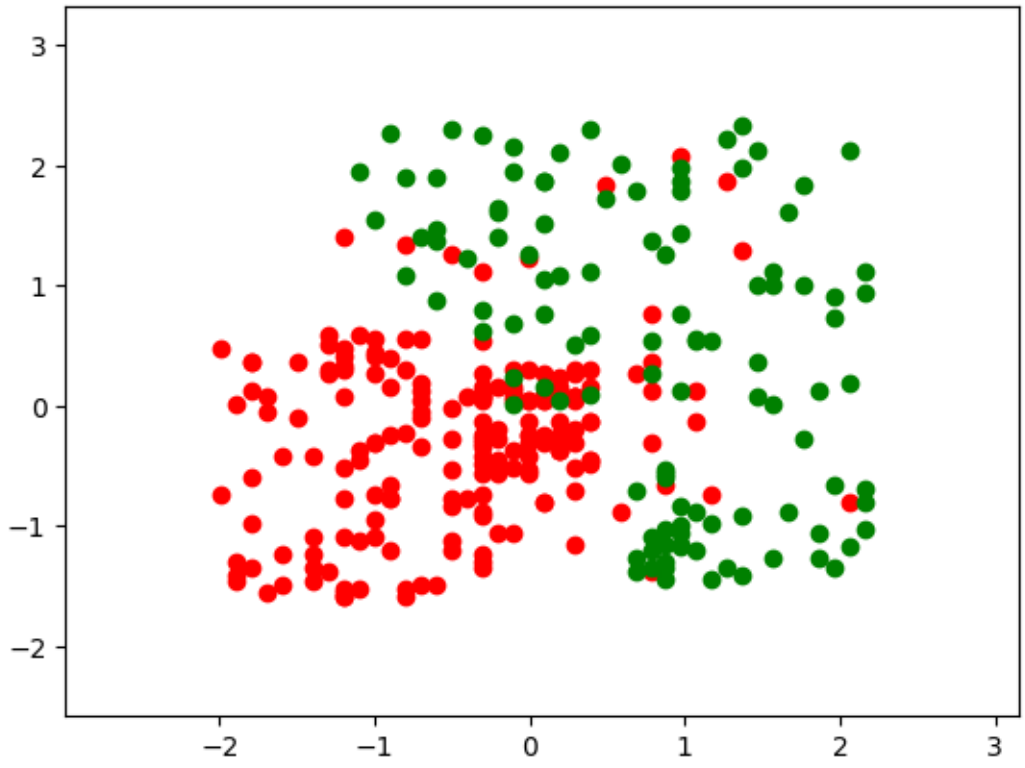
[20]:
```
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))
```

[20]: <matplotlib.contour.QuadContourSet at 0x1b6ef697c50>



[21]:
```
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],  c =␣
 ↪ListedColormap(('red', 'green'))(i), label = j)
```

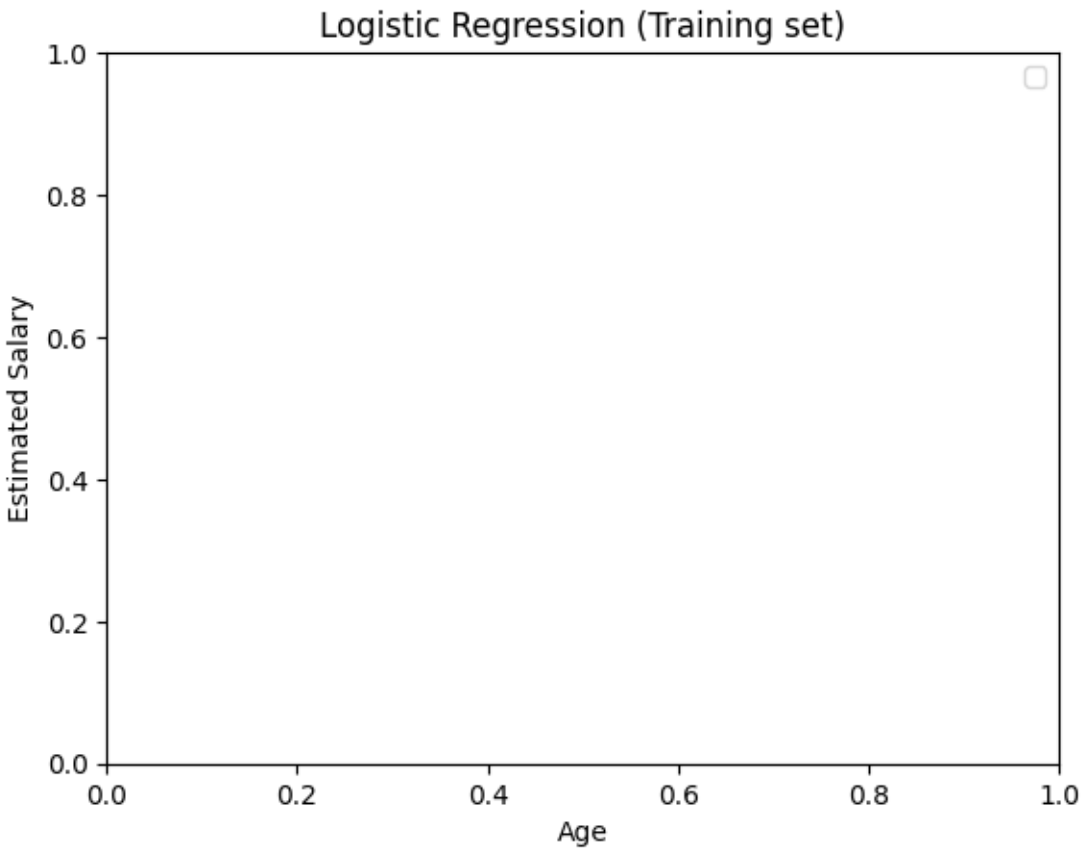
```
C:\TEMP\ipykernel_11804\2800562161.py:4: UserWarning: *c* argument looks like a
single numeric RGB or RGBA sequence, which should be avoided as value-mapping
will have precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if you intend
to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],  c =
ListedColormap(('red', 'green'))(i), label = j)
```



[22]:
```
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```

[22]: `<matplotlib.legend.Legend at 0x1b6ef6eb6e0>`


Logistic Regression (Training set)
Estimated Salary
Age
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

q3840fvvd

April 17, 2024

import pandas as pd from matplotlib import pyplot as plt

[25]:
```
df=pd.read_csv("iris.csv")
df.head(10)
```

[25]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
5	5.4	3.9	1.7	0.4	Setosa
6	4.6	3.4	1.4	0.3	Setosa
7	5.0	3.4	1.5	0.2	Setosa
8	4.4	2.9	1.4	0.2	Setosa
9	4.9	3.1	1.5	0.1	Setosa

[26]:
```
x=df.iloc[:,0:4]
y=df.iloc[:,-1]
y
```

[26]:
```
0         Setosa
1         Setosa
2         Setosa
3         Setosa
4         Setosa
          ...
145    Virginica
146    Virginica
147    Virginica
148    Virginica
149    Virginica
Name: variety, Length: 150, dtype: object
```

[30]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.
 ↪8,random_state=1)
x_test
```

[30]:

	sepal.length	sepal.width	petal.length	petal.width
14	5.8	4.0	1.2	0.2
98	5.1	2.5	3.0	1.1
75	6.6	3.0	4.4	1.4
16	5.4	3.9	1.3	0.4
131	7.9	3.8	6.4	2.0
56	6.3	3.3	4.7	1.6
141	6.9	3.1	5.1	2.3
44	5.1	3.8	1.9	0.4
29	4.7	3.2	1.6	0.2
120	6.9	3.2	5.7	2.3
94	5.6	2.7	4.2	1.3
5	5.4	3.9	1.7	0.4
102	7.1	3.0	5.9	2.1
51	6.4	3.2	4.5	1.5
78	6.0	2.9	4.5	1.5
42	4.4	3.2	1.3	0.2
92	5.8	2.6	4.0	1.2
66	5.6	3.0	4.5	1.5
31	5.4	3.4	1.5	0.4
35	5.0	3.2	1.2	0.2
90	5.5	2.6	4.4	1.2
84	5.4	3.0	4.5	1.5
77	6.7	3.0	5.0	1.7
40	5.0	3.5	1.3	0.3
125	7.2	3.2	6.0	1.8
99	5.7	2.8	4.1	1.3
33	5.5	4.2	1.4	0.2
19	5.1	3.8	1.5	0.3
73	6.1	2.8	4.7	1.2
146	6.3	2.5	5.0	1.9

[33]:
```
from sklearn.preprocessing import LabelEncoder
la_object=LabelEncoder()
y=la_object.fit_transform(y)
y
```

[33]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

[36]:
```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
```

```
model.fit(x_train,y_train)
```

[36]:
```
GaussianNB()
```

[38]:
```
y_predicted=model.predict(x_test)
```

[39]:
```
y_predicted
```

[39]:
```
array(['Setosa', 'Versicolor', 'Versicolor', 'Setosa', 'Virginica',
       'Versicolor', 'Virginica', 'Setosa', 'Setosa', 'Virginica',
       'Versicolor', 'Setosa', 'Virginica', 'Versicolor', 'Versicolor',
       'Setosa', 'Versicolor', 'Versicolor', 'Setosa', 'Setosa',
       'Versicolor', 'Versicolor', 'Virginica', 'Setosa', 'Virginica',
       'Versicolor', 'Setosa', 'Setosa', 'Versicolor', 'Virginica'],
      dtype='<U10')
```

[41]:
```
model.score(x_test,y_test)
```

[41]:
```
0.9666666666666667
```

[42]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_predicted)
```

[43]:
```
cm
```

[43]:
```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]], dtype=int64)
```

[45]:
```
cl_report=classification_report(y_test,y_predicted)
```

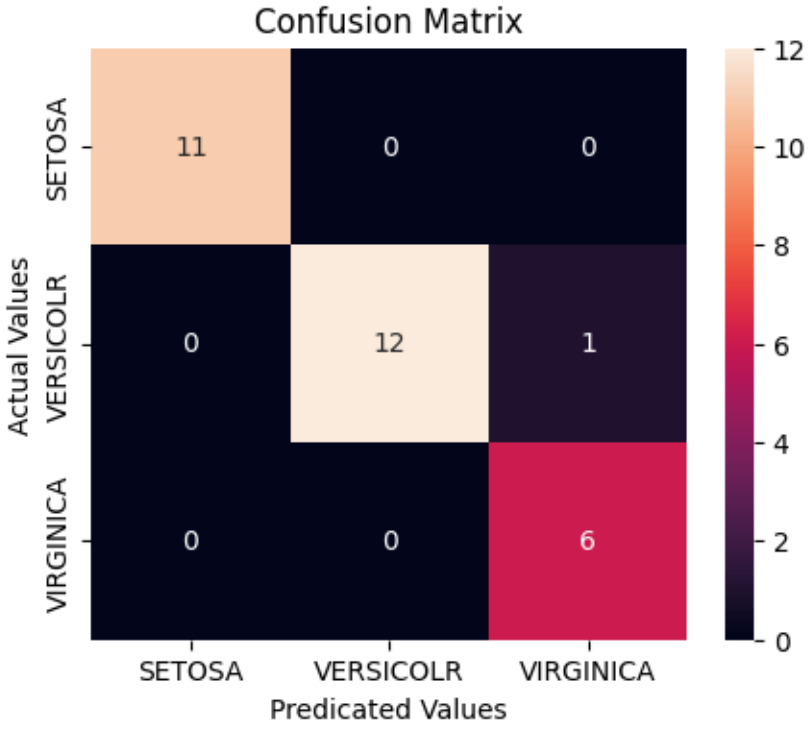
[46]:
```
cl_report
```

[46]:
```
'              precision    recall  f1-score   support\n\n      Setosa
1.00      1.00      1.00        11\n  Versicolor       1.00      0.92      0.96
13\n   Virginica       0.86      1.00      0.92         6\n\n    accuracy
0.97        30\n   macro avg       0.95      0.97      0.96        30\nweighted
avg       0.97      0.97      0.97        30\n'
```

[47]:
```
cm_df=pd.DataFrame(cm,
                   index=['SETOSA','VERSICOLR','VIRGINICA'],
                   columns=['SETOSA','VERSICOLR','VIRGINICA'])
```

[57]:
```
from matplotlib import pyplot as plt
plt.figure(figsize=(5,4))
sns.heatmap(cm_df,annot=True)
plt.title('Confusion Matrix')
```

```
plt.ylabel('Actual Values')
plt.xlabel('Predicated Values')
plt.show()
```



[58]:
```
def accuracy_cm(tp,fn,fp,tn):
    return (tp+tn)/(tp+fp+tn+fn)

def precision_cm(tp,fn,fp,tn):
    return tp/(tp+fp)

def recall_cm(tp,fn,fp,tn):
    return tp/(tp+fn)

def f1_score(tp,fn,fp,tn):
    return (2/(1/recall_cm(tp,fn,fp,tn))+precision_cm(tp,fn,fp,tn))

def error_rate_cm(tp,fn,fp,tn):
    return 1-accuracy_cm(tp,fn,fp,tn)
```

[59]:
```
tp = cm[2][2]
fn = cm[2][0]+cm[2][1]
fp = cm[0][2]+cm[1][2]
tn = cm[0][0]+cm[0][1]+cm[1][0]+cm[1][1]
```

```
print("For Virginica \n")
print("Accuracy    :",accuracy_cm(tp,fn,fp,tn))
print("Precision   :",precision_cm(tp,fn,fp,tn))
print("Recall      :",recall_cm(tp,fn,fp,tn))
print("F1-Score    :",f1_score(tp,fn,fp,tn))
print("Error rate  :",error_rate_cm(tp,fn,fp,tn))
```

```
For Virginica

Accuracy    : 0.9666666666666667
Precision   : 0.8571428571428571
Recall      : 1.0
F1-Score    : 2.857142857142857
Error rate  : 0.033333333333333326
```

xqncbypd7

April 17, 2024

1. Extract Sample Document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Steaming and Lemmatization

[1]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[1]:
```
True
```

Tokenization

[2]:
```
from nltk import word_tokenize, sent_tokenize
sent="Sachin is considered to be one of the greatest cricket player. Virat is␣
 ↪the captain of tthe Indian cricket team"
print(word_tokenize(sent))
print(sent_tokenize(sent))
```

```
['Sachin', 'is', 'considered', 'to', 'be', 'one', 'of', 'the', 'greatest',
'cricket', 'player', '.', 'Virat', 'is', 'the', 'captain', 'of', 'tthe',
'Indian', 'cricket', 'team']
['Sachin is considered to be one of the greatest cricket player.', 'Virat is the
captain of tthe Indian cricket team']
```

Stop Wrods Removal

[3]:
```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stop_words=stopwords.words('english')
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data…
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's",
'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what',
```

```
'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is',
'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some',
'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',
"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]

[nltk_data]   Package stopwords is already up-to-date!
```

[4]:
```
token=word_tokenize(sent)
cleaned_token=[]
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
print("This is the unclean version: ",token)
print("This is the cleaned version: ",cleaned_token)
```

```
This is the unclean version:  ['Sachin', 'is', 'considered', 'to', 'be', 'one',
'of', 'the', 'greatest', 'cricket', 'player', '.', 'Virat', 'is', 'the',
'captain', 'of', 'tthe', 'Indian', 'cricket', 'team']
This is the cleaned version:  ['Sachin', 'considered', 'one', 'greatest',
'cricket', 'player', '.', 'Virat', 'captain', 'tthe', 'Indian', 'cricket',
'team']
```

[5]:
```
words=[cleaned_token.lower() for cleaned_token in cleaned_token if␣
 ↪cleaned_token.isalpha()]
```

[6]:
```
print(words)
```

```
['sachin', 'considered', 'one', 'greatest', 'cricket', 'player', 'virat',
'captain', 'tthe', 'indian', 'cricket', 'team']
```

Steamming

[7]:
```
from nltk.stem import PorterStemmer
stemmer=PorterStemmer()
port_stemmer_output=[stemmer.stem(words) for words in words]
print(port_stemmer_output)
```

```
['sachin', 'consid', 'one', 'greatest', 'cricket', 'player', 'virat', 'captain',
'tthe', 'indian', 'cricket', 'team']
```

Lemmatization

[35]:
```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer=WordNetLemmatizer()
lemmatizer_output = [lemmatizer.lemmatize(word) for word in words]
print(lemmatizer_output)
```

```
['sachin', 'considered', 'one', 'greatest', 'cricket', 'player', 'virat',
'captain', 'tthe', 'indian', 'cricket', 'team']
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

POS Tagging

[36]:
```
from nltk import pos_tag
import nltk
nltk.download('averaged_perceptron_tagger')
token=word_tokenize(sent)
cleaned_token=[]
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
tagged=pos_tag(cleaned_token)
print(tagged)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
[('Sachin', 'NNP'), ('considered', 'VBD'), ('one', 'CD'), ('greatest', 'JJS'),
('cricket', 'NN'), ('player', 'NN'), ('.', '.'), ('Virat', 'NNP'), ('captain',
'VBP'), ('tthe', 'JJ'), ('Indian', 'JJ'), ('cricket', 'NN'), ('team', 'NN')]
```

2. Create Representation of document by calculating Term Frequency and Inverse Doucment Frequency

[16]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
```

[18]:
```
docs=["Sachine is considered to be one of the greatest cricket players",
      "Federer is considered one of the greatest tennis players",
      "Nadal is considered one of the greatest tennis players",
      "Virat is the captain of the India Cricekt Team"]
```

[37]:
```
vectorizer=TfidfVectorizer(analyzer="word", norm=None, use_idf=True, smooth_i␣
↪df=True)
Mat=vectorizer.fit(docs)
print(Mat.vocabulary_)
```

```
{'sachine': 13, 'is': 8, 'considered': 2, 'to': 17, 'be': 0, 'one': 11, 'of':
10, 'the': 16, 'greatest': 6, 'cricket': 4, 'players': 12, 'federer': 5,
'tennis': 15, 'nadal': 9, 'virat': 18, 'captain': 1, 'india': 7, 'cricekt': 3,
'team': 14}
```

[38]:
```
vectorizer = TfidfVectorizer()
tfidfMat = vectorizer.fit_transform(docs)
```

[39]:
```
print(tfidfMat)
```

```
  (0, 12)       0.25139160418660417
  (0, 4)        0.39385352655962536
  (0, 6)        0.25139160418660417
  (0, 16)       0.20552910892306617
  (0, 10)       0.20552910892306617
  (0, 11)       0.25139160418660417
  (0, 0)        0.39385352655962536
  (0, 17)       0.39385352655962536
  (0, 2)        0.25139160418660417
  (0, 8)        0.20552910892306617
  (0, 13)       0.39385352655962536
  (1, 15)       0.39088800650868233
  (1, 5)        0.4957918718681808
  (1, 12)       0.3164575295297054
  (1, 6)        0.3164575295297054
  (1, 16)       0.258724766352802
  (1, 10)       0.258724766352802
  (1, 11)       0.3164575295297054
  (1, 2)        0.3164575295297054
  (1, 8)        0.258724766352802
  (2, 9)        0.4957918718681808
  (2, 15)       0.39088800650868233
  (2, 12)       0.3164575295297054
  (2, 6)        0.3164575295297054
  (2, 16)       0.258724766352802
  (2, 10)       0.258724766352802
  (2, 11)       0.3164575295297054
  (2, 2)        0.3164575295297054
  (2, 8)        0.258724766352802
  (3, 14)       0.3882533152281991
  (3, 3)        0.3882533152281991
  (3, 7)        0.3882533152281991
```

```
  (3, 1)        0.3882533152281991
  (3, 18)       0.3882533152281991
  (3, 16)       0.40521337265821117
  (3, 10)       0.20260668632910558
  (3, 8)        0.20260668632910558
```

[41]:
```
features_name=vectorizer.get_feature_names_out()
print(features_name)
```

```
['be' 'captain' 'considered' 'cricekt' 'cricket' 'federer' 'greatest'
 'india' 'is' 'nadal' 'of' 'one' 'players' 'sachine' 'team' 'tennis' 'the'
 'to' 'virat']
```

[43]:
```
dense=tfidfMat.todense()
denselist=dense.tolist()
df=pd.DataFrame(denselist,columns=features_name)
```

[44]:
```
df
```

[44]:

	be	captain	considered	cricekt	cricket	federer	greatest	\
0	0.393854	0.000000	0.251392	0.000000	0.393854	0.000000	0.251392	
1	0.000000	0.000000	0.316458	0.000000	0.000000	0.495792	0.316458	
2	0.000000	0.000000	0.316458	0.000000	0.000000	0.000000	0.316458	
3	0.000000	0.388253	0.000000	0.388253	0.000000	0.000000	0.000000	

	india	is	nadal	of	one	players	sachine	\
0	0.000000	0.205529	0.000000	0.205529	0.251392	0.251392	0.393854	
1	0.000000	0.258725	0.000000	0.258725	0.316458	0.316458	0.000000	
2	0.000000	0.258725	0.495792	0.258725	0.316458	0.316458	0.000000	
3	0.388253	0.202607	0.000000	0.202607	0.000000	0.000000	0.000000	

	team	tennis	the	to	virat
0	0.000000	0.000000	0.205529	0.393854	0.000000
1	0.000000	0.390888	0.258725	0.000000	0.000000
2	0.000000	0.390888	0.258725	0.000000	0.000000
3	0.388253	0.000000	0.405213	0.000000	0.388253

[51]:
```
features_name = sorted(vectorizer.get_feature_names_out())
```

[54]:
```
docList=['Doc 1','Doc 2','Doc 3','Doc 4']
skDocsIfIdfdf=pd.DataFrame(tfidfMat.todense(),index=sorted(docList),␣
 ↪columns=features_name)
print(skDocsIfIdfdf)
```

	be	captain	considered	cricekt	cricket	federer	greatest	\
Doc 1	0.393854	0.000000	0.251392	0.000000	0.393854	0.000000	0.251392	
Doc 2	0.000000	0.000000	0.316458	0.000000	0.000000	0.495792	0.316458	
Doc 3	0.000000	0.000000	0.316458	0.000000	0.000000	0.000000	0.316458	

```
Doc 4  0.000000  0.388253    0.000000  0.388253  0.000000  0.000000  0.000000

          india        is     nadal        of       one   players   sachine  \
Doc 1  0.000000  0.205529  0.000000  0.205529  0.251392  0.251392  0.393854
Doc 2  0.000000  0.258725  0.000000  0.258725  0.316458  0.316458  0.000000
Doc 3  0.000000  0.258725  0.495792  0.258725  0.316458  0.316458  0.000000
Doc 4  0.388253  0.202607  0.000000  0.202607  0.000000  0.000000  0.000000

           team    tennis       the        to     virat
Doc 1  0.000000  0.000000  0.205529  0.393854  0.000000
Doc 2  0.000000  0.390888  0.258725  0.000000  0.000000
Doc 3  0.000000  0.390888  0.258725  0.000000  0.000000
Doc 4  0.388253  0.000000  0.405213  0.000000  0.388253
```

[55]:
```
csim=cosine_similarity(tfidfMat,tfidfMat)
```

[57]:
```
csimDf = pd.DataFrame(csim, index=sorted(docList), columns=sorted(docList))
```

[58]:
```
print(csimDf)
```

```
          Doc 1     Doc 2     Doc 3     Doc 4
Doc 1  1.000000  0.477745  0.477745  0.166566
Doc 2  0.477745  1.000000  0.754190  0.209677
Doc 3  0.477745  0.754190  1.000000  0.209677
Doc 4  0.166566  0.209677  0.209677  1.000000
```

nlpwqgty6

April 17, 2024

[6]:
```
import seaborn as sns
import pandas as pd
titanic=sns.load_dataset("titanic")
titanic
```

[6]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	
..	…	…	…	…	…	…	…	…	…	
886	0	2	male	27.0	0	0	13.0000	S	Second	
887	1	1	female	19.0	0	0	30.0000	S	First	
888	0	3	female	NaN	1	2	23.4500	S	Third	
889	1	1	male	26.0	0	0	30.0000	C	First	
890	0	3	male	32.0	0	0	7.7500	Q	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True
..	…	…	…	…	…	…
886	man	True	NaN	Southampton	no	True
887	woman	False	B	Southampton	yes	True
888	woman	False	NaN	Southampton	no	False
889	man	True	C	Cherbourg	yes	True
890	man	True	NaN	Queenstown	no	True

[891 rows x 15 columns]

[7]:
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
```

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

[8]:
```
x=titanic["fare"]
x
```

[8]:
```
0       7.2500
1      71.2833
2       7.9250
3      53.1000
4       8.0500
        …
886    13.0000
887    30.0000
888    23.4500
889    30.0000
890     7.7500
Name: fare, Length: 891, dtype: float64
```

[9]:
```
titanic.describe()
```

[9]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Data Cleanup

[10]:
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

[24]:
```
titanic_cleaned = titanic.drop(['pclass', 'embarked', 'deck', 'embark_town'],␣
↪axis=1)
titanic_cleaned.head(15)
```

[24]:

	survived	sex	age	sibsp	parch	fare	class	who	adult_male
0	0	male	22.0	1	0	7.2500	Third	man	True
1	1	female	38.0	1	0	71.2833	First	woman	False
2	1	female	26.0	0	0	7.9250	Third	woman	False
3	1	female	35.0	1	0	53.1000	First	woman	False
4	0	male	35.0	0	0	8.0500	Third	man	True
5	0	male	NaN	0	0	8.4583	Third	man	True
6	0	male	54.0	0	0	51.8625	First	man	True
7	0	male	2.0	3	1	21.0750	Third	child	False
8	1	female	27.0	0	2	11.1333	Third	woman	False
9	1	female	14.0	1	0	30.0708	Second	child	False
10	1	female	4.0	1	1	16.7000	Third	child	False
11	1	female	58.0	0	0	26.5500	First	woman	False
12	0	male	20.0	0	0	8.0500	Third	man	True
13	0	male	39.0	1	5	31.2750	Third	man	True
14	0	female	14.0	0	0	7.8542	Third	child	False

```
   alive  alone
0     no  False
1    yes  False
2    yes   True
3    yes  False
4     no   True
5     no   True
6     no   True
7     no  False
8    yes  False
9    yes  False
10   yes  False
11   yes   True
12    no   True
13    no  False
14    no   True
```

[25]:
```
titanic_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   survived    891 non-null    int64
 1   sex         891 non-null    object
 2   age         714 non-null    float64
 3   sibsp       891 non-null    int64
 4   parch       891 non-null    int64
 5   fare        891 non-null    float64
 6   class       891 non-null    category
 7   who         891 non-null    object
 8   adult_male  891 non-null    bool
 9   alive       891 non-null    object
 10  alone       891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(3), object(3)
memory usage: 58.6+ KB
```

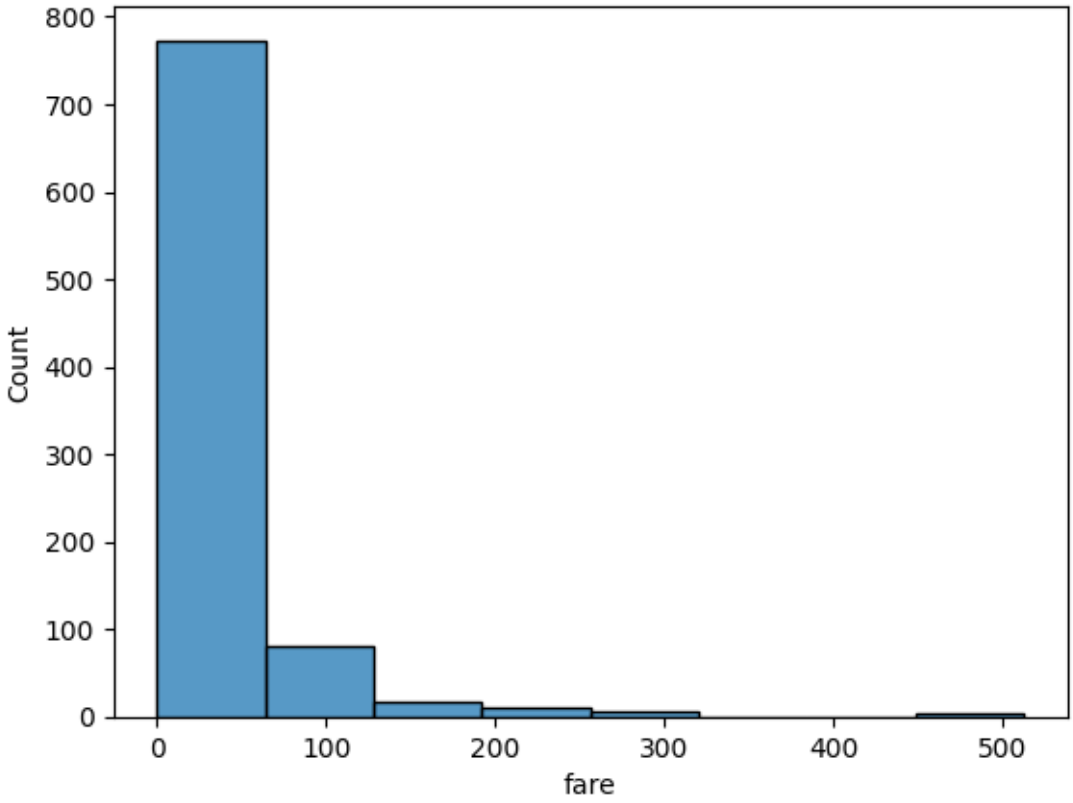
[26]:
```
titanic_cleaned.isnull().sum()
```

[26]:
```
survived       0
sex            0
age          177
sibsp          0
parch          0
fare           0
class          0
```

```
who             0
adult_male      0
alive           0
alone           0
dtype: int64
```

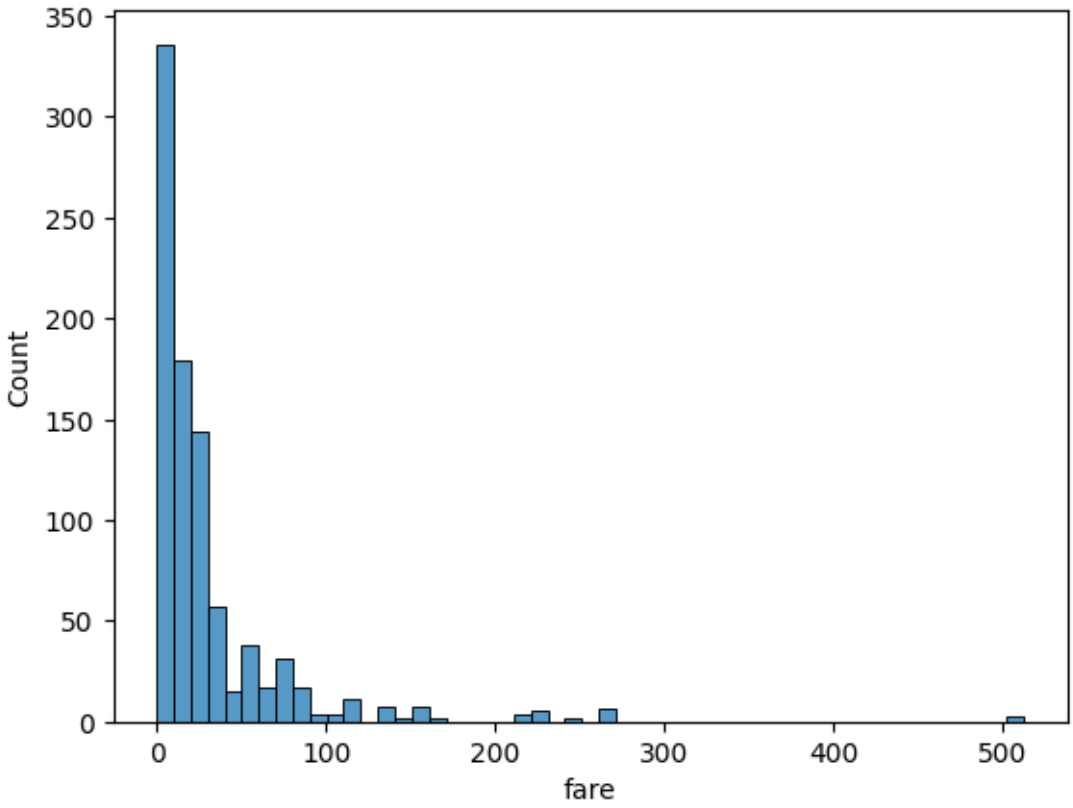
[31]:
```
sns.histplot(data=titanic,x="fare",bins=8)
```

[31]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



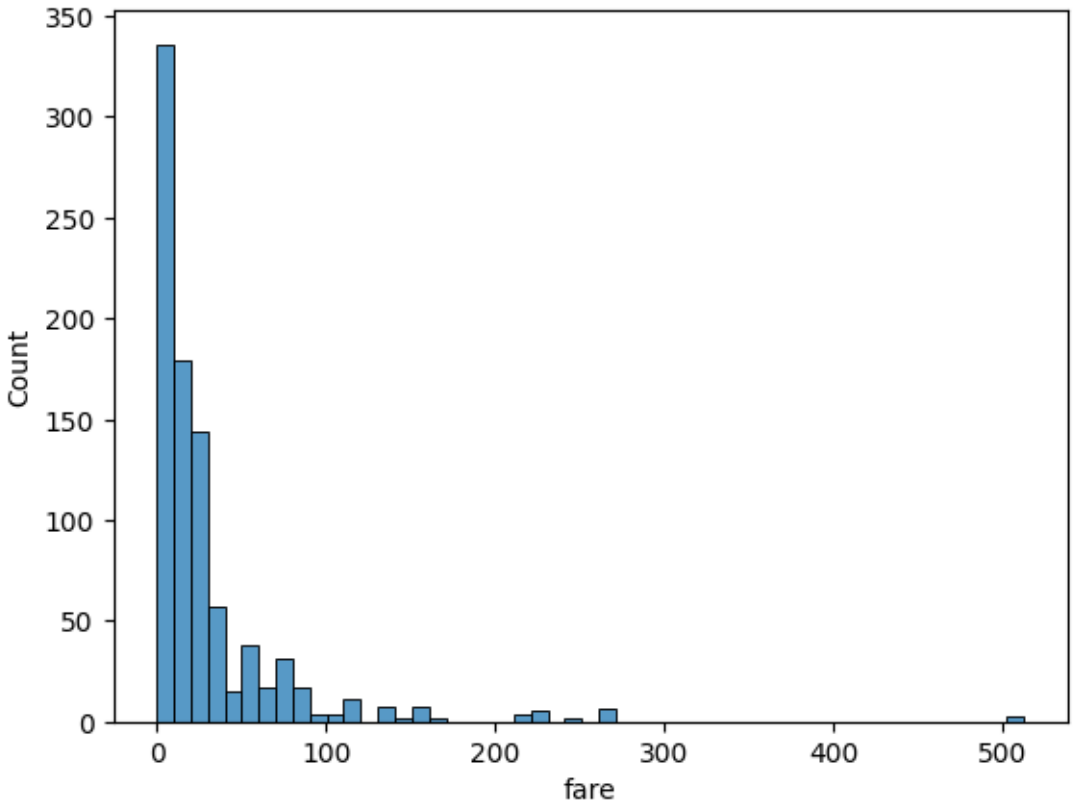
[17]:
```
sns.histplot(data=titanic,x="fare",binwidth=10)
```

[17]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



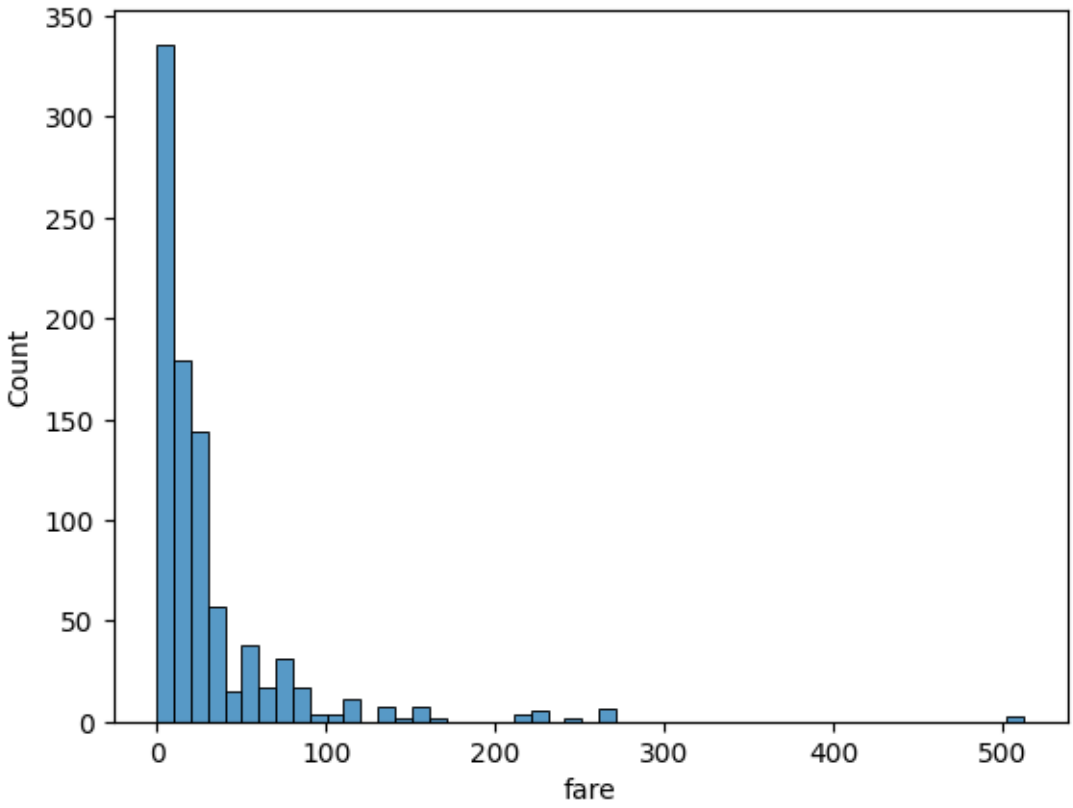
[19]:
```
sns.histplot(data=titanic,x="fare",bins=20,binwidth=10)
```

[19]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



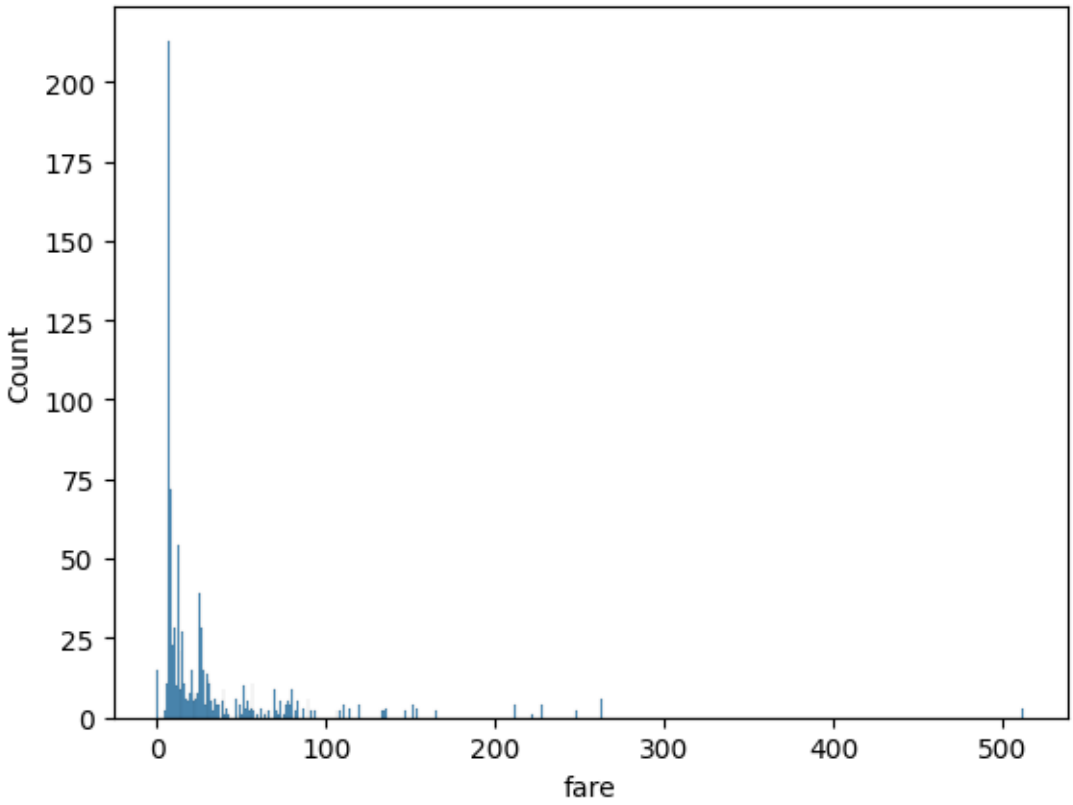
[20]:
```
sns.histplot(data=titanic,x="fare",binwidth=10)
```

[20]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



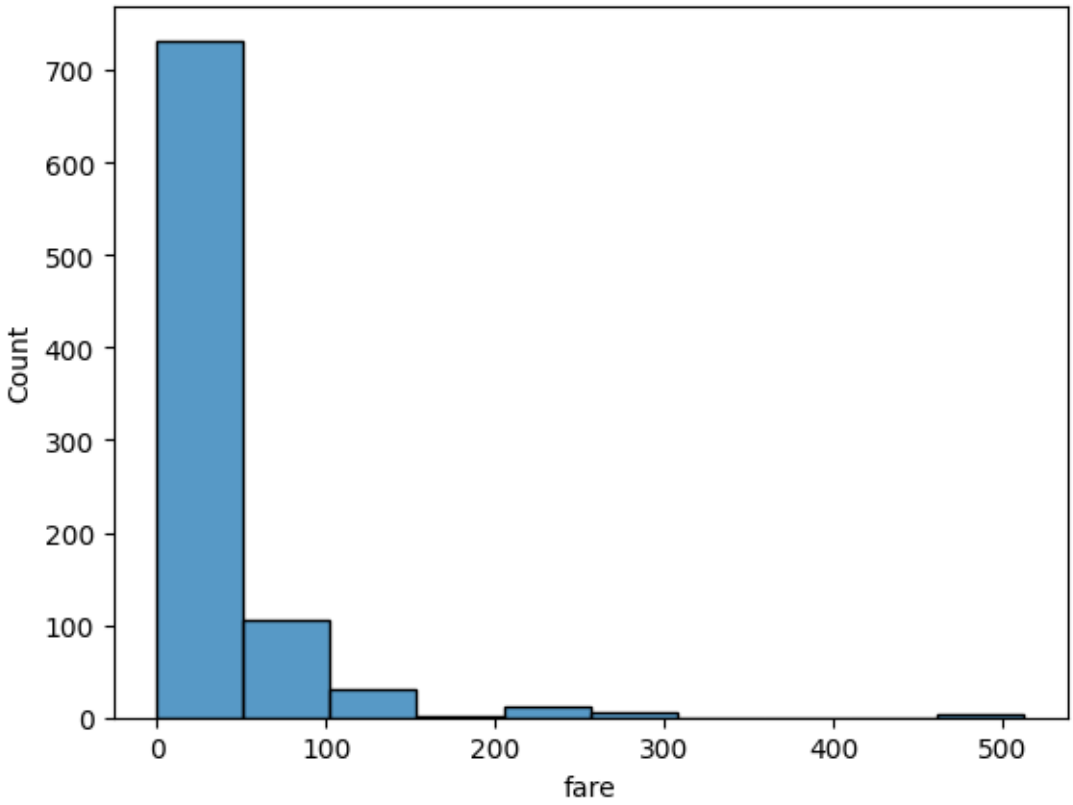
[21]:
```
sns.histplot(data=titanic,x="fare",bins=20,binwidth=1)
```

[21]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



[22]:
```
sns.histplot(data=titanic,x="fare",bins=20,binwidth=50)
```

[22]:
```
<Axes: xlabel='fare', ylabel='Count'>
```


Count
700
600
500
400
300
200
100
0
0
100
200
300
400
500
fare

yjkaiv60j

April 17, 2024

[1]:
```
import seaborn as sns
titanic=sns.load_dataset("titanic")
titanic
```

[1]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

[2]:
```
titanic.head(10)
```

[2]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0         0       3    male  22.0      1      0   7.2500        S   Third
1         1       1  female  38.0      1      0  71.2833        C   First
```

```
2         1       3  female  26.0      0      0   7.9250        S   Third
3         1       1  female  35.0      1      0  53.1000        S   First
4         0       3    male  35.0      0      0   8.0500        S   Third
5         0       3    male   NaN      0      0   8.4583        Q   Third
6         0       1    male  54.0      0      0  51.8625        S   First
7         0       3    male   2.0      3      1  21.0750        S   Third
8         1       3  female  27.0      0      2  11.1333        S   Third
9         1       2  female  14.0      1      0  30.0708        C  Second

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
5    man        True  NaN   Queenstown    no   True
6    man        True    E  Southampton    no   True
7  child       False  NaN  Southampton    no  False
8  woman       False  NaN  Southampton   yes  False
9  child       False  NaN    Cherbourg   yes  False
```

[3]:
```
titanic.head(10)
```

[3]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0         0       3    male  22.0      1      0   7.2500        S   Third
1         1       1  female  38.0      1      0  71.2833        C   First
2         1       3  female  26.0      0      0   7.9250        S   Third
3         1       1  female  35.0      1      0  53.1000        S   First
4         0       3    male  35.0      0      0   8.0500        S   Third
5         0       3    male   NaN      0      0   8.4583        Q   Third
6         0       1    male  54.0      0      0  51.8625        S   First
7         0       3    male   2.0      3      1  21.0750        S   Third
8         1       3  female  27.0      0      2  11.1333        S   Third
9         1       2  female  14.0      1      0  30.0708        C  Second

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
5    man        True  NaN   Queenstown    no   True
6    man        True    E  Southampton    no   True
7  child       False  NaN  Southampton    no  False
8  woman       False  NaN  Southampton   yes  False
9  child       False  NaN    Cherbourg   yes  False
```

[4]:
```
titanic.describe()
```

[4]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

[5]:
```
titanic.describe()
```

[5]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

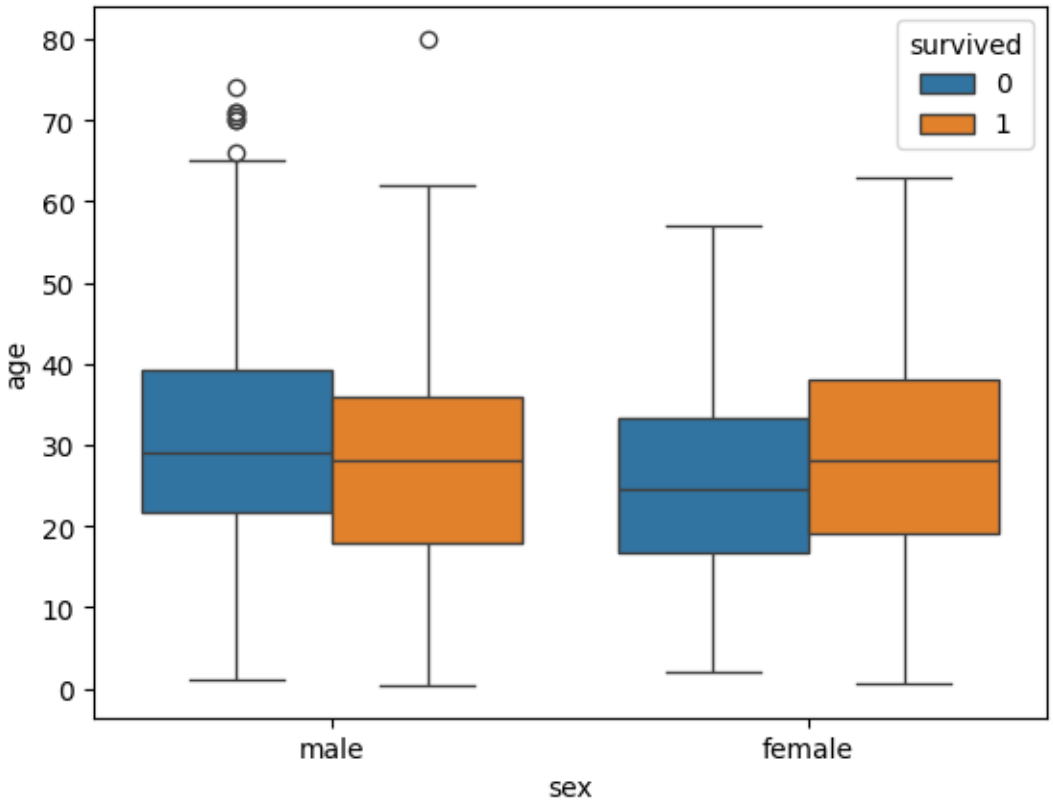
[6]:
```
titanic.describe()
```

[6]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

[7]:
```
sns.boxplot(x="sex",y="age",data=titanic,hue="survived")
```

[7]:
```
<Axes: xlabel='sex', ylabel='age'>
```


survived
0
1
80
70
60
50
40
30
20
10
0
age
male
female
sex

practical10

April 19, 2024

[28]:
```
import numpy as np
import pandas as pd
```

[45]:
```
df = pd.read_csv("Iris1.csv",header=None)
```

[46]:
```
df.head()
```

[46]:

	0	1	2	3	4	5
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Q1. How many features are there and what are their types?

[47]:
```
column = len(list(df))
column
```

[47]: 6

[48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       150 non-null    int64
 1   1       150 non-null    float64
 2   2       150 non-null    float64
 3   3       150 non-null    float64
 4   4       150 non-null    float64
 5   5       150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

[49]:
```
np.unique(df[5])
```

[49]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Q2. Compute and display summary statistics for each feature available in the dataset.

[50]:
```
df.describe()
```

[50]:

	0	1	2	3	4
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Q3. Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram.

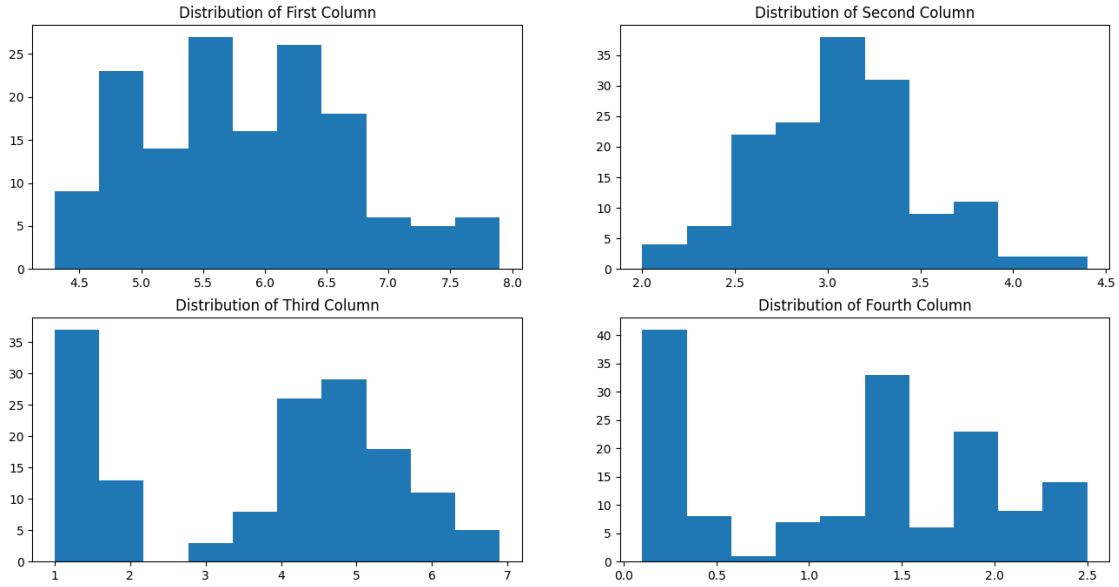
[51]:
```
fig, axes = plt.subplots(2, 2, figsize=(16, 8))


axes[0,0].set_title("Distribution of First Column")
axes[0,0].hist(df[1]);

axes[0,1].set_title("Distribution of Second Column")
axes[0,1].hist(df[2]);

axes[1,0].set_title("Distribution of Third Column")
axes[1,0].hist(df[3]);

axes[1,1].set_title("Distribution of Fourth Column")
axes[1,1].hist(df[4]);
```



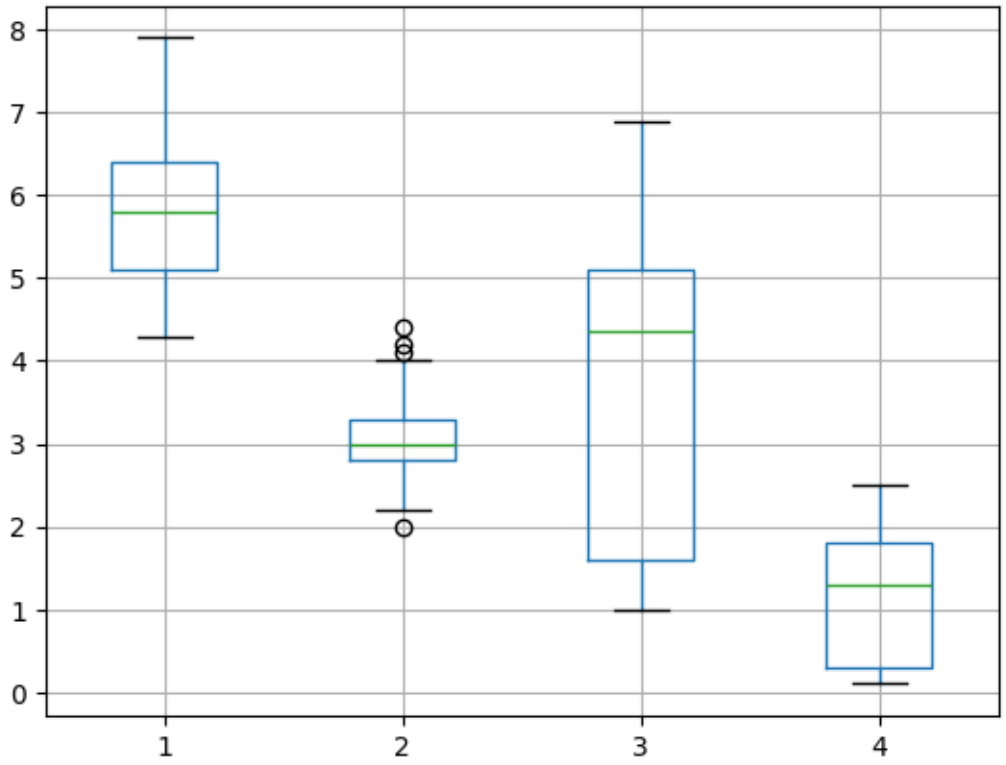
Q4. Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers.

[52]:
```
import matplotlib.pyplot as plt
```

[53]:
```
df.columns
```

[53]:
```
Index([0, 1, 2, 3, 4, 5], dtype='int64')
```

[54]:
```
x=df[[1, 2, 3, 4]]
x.boxplot()
plt.show()
```


8
7
6
5
4
3
2
1
0
1
2
3
4

[ ]: